

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to:

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

PATENT

Attorney Docket No.: 021735-000400US

On _____

TOWNSEND and TOWNSEND and CREW LLP

By: _____

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:

Craig Weissman et al.

Application No.: 10/669,523

Filed: September 23, 2003

For: QUERY OPTIMIZATION IN A
MULTI-TENANT DATABASE
SYSTEM

Customer No.: 20350

Confirmation No. 5140

Examiner: Khanh B. Pham

Technology Center/Art Unit: 2166

DECLARATION UNDER 37 C.F.R. 1.131

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

We, Craig Weissman, Dave Moellenhoff, Simon Wong and Paul Nakada, each hereby declare and state as follows:

1. We, Craig Weissman, Dave Moellenhoff, Simon Wong and Paul Nakada (hereinafter "the SFDC inventors"), are joint inventors of the subject matter recited in all pending claims of the above-entitled application.

2. All the work described in this declaration was performed in the United States of America.

Appl. No. 10/669,523

PATENT

3. Prior to August 29, 2003, the SFDC inventors conceived the subject matter of the inventions described in the pending claims in the United States of America.

4. From before August 29, 2003 to September 23, 2003, the SFDC inventors and Mr. Gerald T. Gray, a patent attorney at Townsend and Townsend and Crew and attorney of record of the above identified application, diligently worked towards filing a patent application in the U.S. Patent and Trademark Office as evidenced by Attachment 1, which is a portion of the billing records showing Mr. Gray's work for the above-referenced patent application.

5. Attachment 1 shows that Mr. Gray had begun drafting the application prior to August 29, 2003, and that he had sent a draft application by e-mail for review on August 19, 2003, prior to August 29, 2003.

5.1 Attachment 2 is a copy of the e-mail of August 19, 2003 from Mr. Gray to Mr. Craig Weissman enclosing a draft application and figures 1-3. The draft application and figures evidences that the SFDC inventors had conceived of the inventions prior to August 19, 2003, which is prior to August 29, 2003.

6. Attachment 1 also shows that Mr. Gray worked 8 days, inclusive, between August 19, 2003 and September 23, 2003, logging 10.7 hours for this matter, including revising the application per comments the SFDC inventors had provided on the draft application.

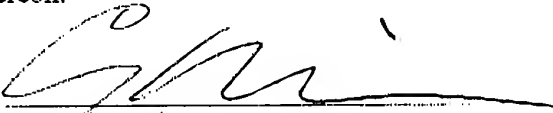
7. On September 23, 2003, the application describing the SFDC inventors' inventions was filed in the U.S. Patent and Trademark Office. Attachment 1 reflects that Mr. Gray worked 1.4 hours on September 23, 2003 to prepare and file the application and an Information Disclosure Statement (IDS).

8. We each declare, individually, that all statements made herein of our own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false

Appl. No. 10/669,523

PATENT

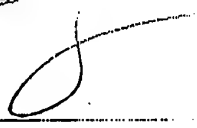
statements, and the like, so made are punishable by a fine or imprisonment, or both, under §1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Date: 4/2/07By: 
Craig Weissman

Date: _____

By: _____
Dave MoellenhoffDate: Apr 2, 2007By: 
Simon Wong

Date: _____

By: 
Paul Nakada

Enclosures:

Attachment 1

Attachment 2

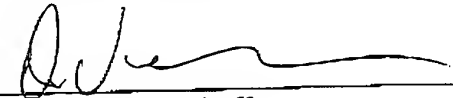
61020012 v1

statements, and the like, so made are punishable by a fine or imprisonment, or both, under §1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Date: _____

By: _____
Craig Weissman

Date: 4-4-07 _____

By:  _____
Dave Moellenhoff

Date: _____

By: _____
Simon Wong

Date: _____

By: _____
Paul Nakada

Enclosures:

Attachment 1

Attachment 2

ATTACHMENT 1

Timesheet for Gray, Gerald T. (0292)
Printed 01/06/2006 at 09:19 am Page 1

Filter : Client : 021735 Matter : 000400

Date Worked : 08/16/2003
Client : 021735 SalesForce.com, Inc.
Matter : 000400 021 Multi-Tenant Database
Timekeeper : 0292
Hours : 2.40 B
Location : 05 Walnut Creek
Dollar Amt :

Status : Released : Sep 3 2003 10:06AM

Continue preparing application: revise application and claims. Revise drawings.

Date Worked : 08/18/2003
Client : 021735 SalesForce.com, Inc.
Matter : 000400 021 Multi-Tenant Database
Timekeeper : 0292
Hours : 1.80 B
Location : 05 Walnut Creek
Dollar Amt :

Status : Released : Sep 3 2003 10:06AM

Continue preparing application: revise detailed description and summary sections.

Date Worked : 08/19/2003
Client : 021735 SalesForce.com, Inc.
Matter : 000400 021 Multi-Tenant Database
Timekeeper : 0292
Hours : 1.40 B
Location : 05 Walnut Creek
Dollar Amt :

Status : Released : Sep 3 2003 8:36AM

Revise application. Scan drawings. Transmit draft application to inventor for review.

Date Worked : 08/22/2003
Client : 021735 SalesForce.com, Inc.
Matter : 000400 021 Multi-Tenant Database
Timekeeper : 0292
Hours : 0.40 B
Location : 05 Walnut Creek
Dollar Amt :

Timesheet for Gray, Gerald T. (0292)
Printed 01/06/2006 at 09:19 am Page 2

Filter : Client : 021735 Matter : 000400

Status :

Released : Sep 3 2003 8:39AM

Identify potential prior art references for inclusion in IDS.

Date Worked : 08/29/2003

Client : 021735

SalesForce.com, Inc.

Matter : 000400

021 Multi-Tenant Database

Timekeeper : 0292

Hours : 3.20 B

Location : 05

Walnut Creek

Dollar Amt :

Status :

Released : Sep 3 2003 10:06AM

Revise application per inventor comments. Draft and revise new claims.

Date Worked : 09/02/2003

Client : 021735

SalesForce.com, Inc.

Matter : 000400

021 Multi-Tenant Database

Timekeeper : 0292

Hours : 0.40 B

Location : 05

Walnut Creek

Dollar Amt :

Status :

Released : Sep 3 2003 10:05AM

Revise application and claims. Transmit same to client for review.

Date Worked : 09/11/2003

Client : 021735

SalesForce.com, Inc.

Matter : 000400

021 Multi-Tenant Database

Timekeeper : 0292

Hours : 0.80 B

Location : 05

Walnut Creek

Dollar Amt :

Status :

Released : Oct 2 2003 9:46AM

Communications with client. Revise application per client's suggestions.

Date Worked : 09/12/2003

Client : 021735

SalesForce.com, Inc.

Matter : 000400

021 Multi-Tenant Database

Timesheet for Gray, Gerald T. (0292)
Printed 01/06/2006 at 09:19 am Page 3

Filter : Client : 021735 Matter : 000400

Timekeeper : 0292
Hours : 0.70 B
Location : 05 Walnut Creek
Dollar Amt :

Status : **Released : Oct 2 2003 9:47AM**

Client correspondence. Attempt to contact inventors. Prepare formal papers. Revise application with inventor address information.

Date Worked : 09/16/2003
Client : 021735 SalesForce.com, Inc.
Matter : 000400 021 Multi-Tenant Database
Timekeeper : 0292
Hours : 0.60 B
Location : 05 Walnut Creek
Dollar Amt :

Status : **Released : Oct 2 2003 9:51AM**

Review and revise formal papers. Client correspondence enclosing same and copy of application to client for execution by inventors, as well as instructions for order of execution of documents..

Date Worked : 09/23/2003
Client : 021735 SalesForce.com, Inc.
Matter : 000400 021 Multi-Tenant Database
Timekeeper : 0292
Hours : 1.40 B
Location : 05 Walnut Creek
Dollar Amt :

Status : **Released : Oct 2 2003 9:59AM**

Review file. Review executed formal papers received from client. Prepare and revise IDS. Attend to filing application with executed formal papers and IDS.

Grand Total : 10 Entries.

Hours : 13.10

End of Report

Time for Gray, Gerald T. (0292) for August, 2003
 Printed Friday 01/06/2006 at 09:18 am

Sun	Mon	Tue	Wed	Thu	Fri	Sat
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16 2.4 R
17	18 1.8 R	19 1.4 R	20	21	22 0.4 R	23
24	25	26	27	28	29 3.2 R	30
31						

Billable : 9.20
 Non-Billable : 0.00
 Released : 9.20
 Un-Released : 0.00
 Total : 9.20

Time for Gray, Gerald T. (0292) for September, 2003
 Printed Friday 01/06/2006 at 09:18 am

Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2 0.4 R	3	4	5	6
7	8	9	10	11 0.8 R	12 0.7 R	13
14	15	16 0.6 R	17	18	19	20
21	22	23 1.4 R	24	25	26	27
28	29	30				

Billable : 3.90
 Non-Billable : 0.00
 Released : 3.90
 Un-Released : 0.00
 Total : 3.90

ATTACHMENT 2

Gray, Gerald T.

From: Gray, Gerald T.
Sent: Tuesday, August 19, 2003 9:22 AM
To: 'Craig Weissman'
Cc: 'Julian Ong'
Subject: Figs 1,2 21735-4.PDF;Multi-Tenant DB Query Optimization Application.DOC;FIG 3 21735-4.DOC

Hello Craig:

Attached is a revised draft of the Query Optimization Patent Application for your review.

[Redacted]

Please call if you have any questions.

Regards,

Jerry

Gerald T. Gray
Townsend and Townsend and Crew
2175 North California Blvd., Suite 625
Walnut Creek, CA 94596
Tel: 925-472-5000
Fax: 925-472-8895

www.townsend.com

Offices in:

Denver | Palo Alto | San Diego | San Francisco | Seattle | Walnut Creek

This message, and any attachments thereto, is intended only for the use of the individual or entity to which it is addressed and may contain information that is privileged, confidential, and/or exempt from disclosure by applicable law or court order. If the reader is not the intended recipient, or the employee or agent responsible for delivering the message to the intended recipient, you are hereby notified that any use, dissemination, distribution, or copying of this message and any attachments thereto are strictly prohibited. If you have received this communication and any attachments thereto (whether directly or indirectly as an embedded communication) in error, please notify us immediately by reply email at the above address. Please also immediately delete the message and any attachments thereto from your computer system. Thank you.

4/2/2007

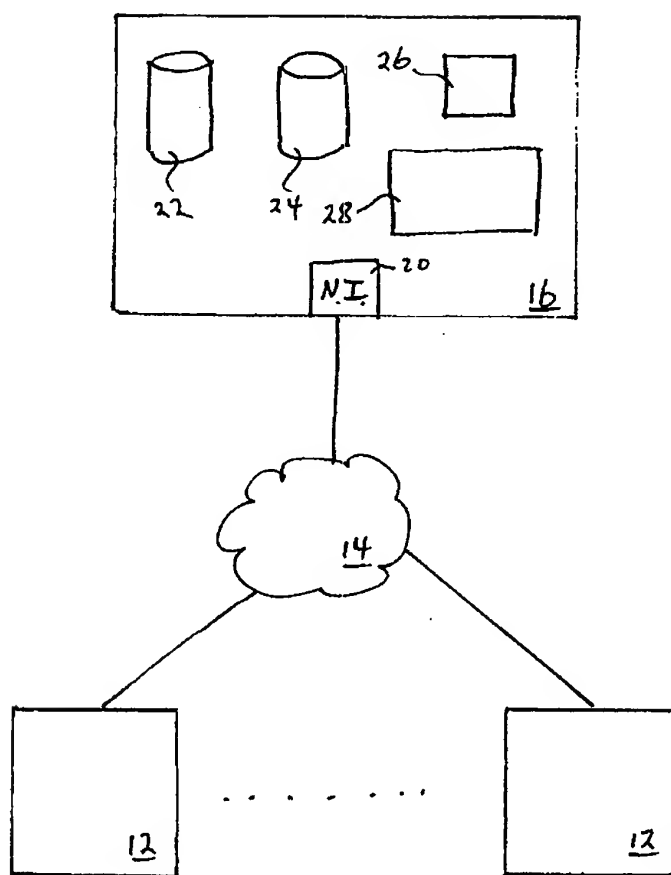


FIG. 1

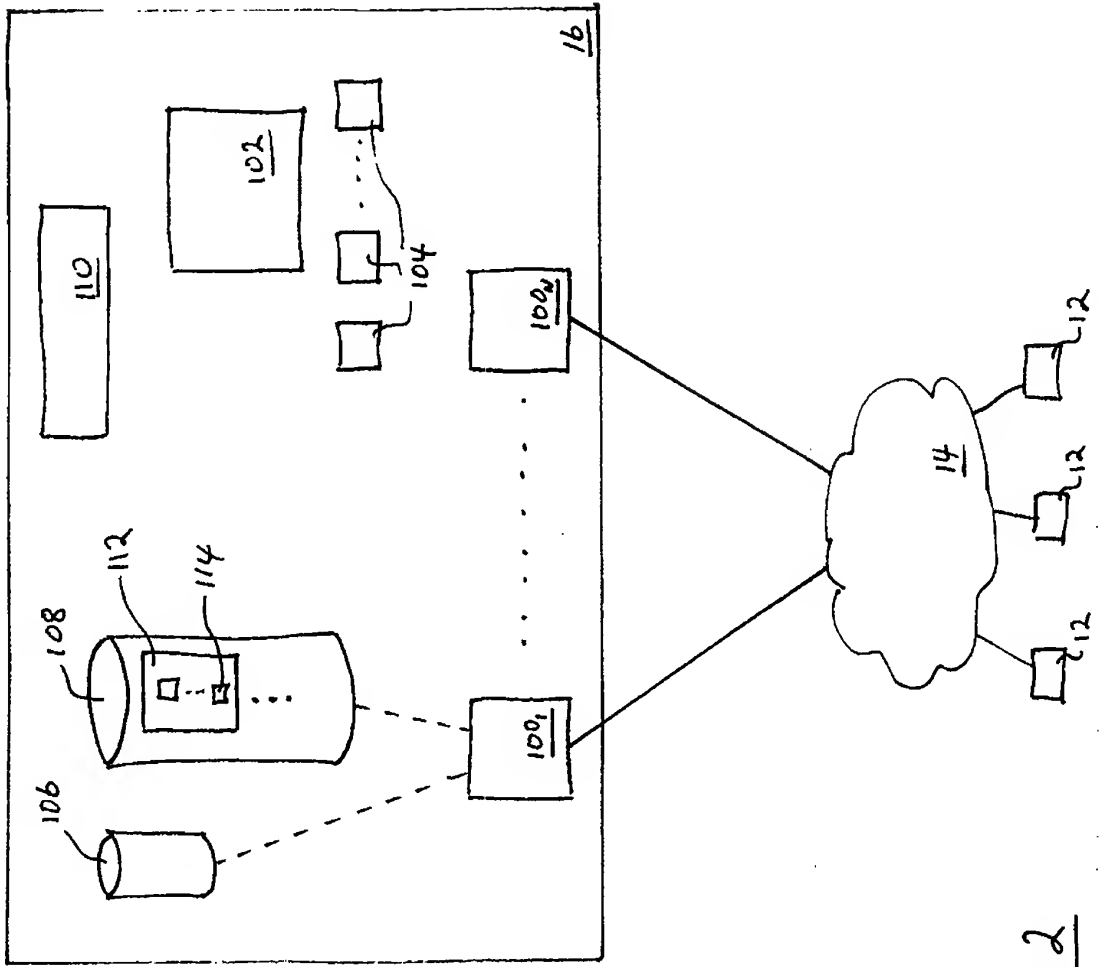


FIG. 2

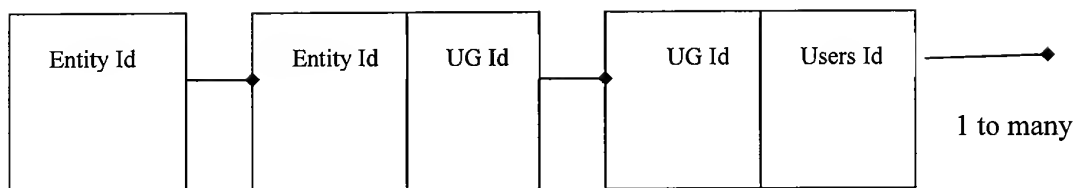


FIG 3.

PATENT APPLICATION

QUERY OPTIMIZATION IN A MULTI-TENANT DATABASE SYSTEM

Inventor: Craig Weissman, a citizen of The United States, residing at

Assignee: Salesforce.com, Inc.
attn: Julian Ong, Assistant General Counsel
The Landmark @ One Market, Suite 300
San Francisco, CA 94105

Entity: Small Business Concern

QUERY OPTIMIZATION IN A MULTI-TENANT DATABASE SYSTEM

BACKGROUND OF THE INVENTION

[0001] The present invention relates generally to database systems and more particularly to query optimization systems and methods for use in multi-tenant database systems, wherein a centralized computer or set of computing devices serve and store applications and data for use by multiple tenants.

[0002] Multi-tenant database systems allow for users to access applications and/or data from a network source that, to the user, appears to be centralized (but might be distributed for backup, redundancy and/or performance reasons). An example of a multi-tenant system is a computing system that is accessible to multiple independent parties to provide those parties with application execution and/or data storage. Where there is an appearance of centralization, and network access, each subscribing party (*e.g.*, a “tenant”) can access the system to perform application functions, including manipulating that tenant’s data.

[0003] With a multi-tenant system, the tenants have the advantage that they need not install software, maintain backups, move data to laptops to provide portability, *etc.* Rather, each tenant user need only be able to access the multi-tenant system to operate the applications and access that tenant’s data. One such system usable for sales force automation is the multi-tenant system accessible to Salesforce.com subscribers. With such systems, a user need only have access to a user system with network connectivity, such as a desktop computer with Internet access and a browser or other HTTP client, or other suitable Internet client.

[0004] In database systems, to access, retrieve and process stored data, a query is generated, automatically or manually, in accordance with the application program interface protocol for the database. In the case of a relational database, the standard protocol is the structured query language (SQL). SQL statements are used both for interactive queries for data from the database and for gathering data and statistics. The efficiency of the query method underlying the actual query is dependent in part on the size and complexity of the data structure scheme of the database and in part on the query logic used.

[0005] Previous multi-tenant database query methods have been inefficient because such methods do not understand, and fail to account for, the unique characteristics of each tenant’s data. For example, while one tenant’s data may include numerous short records having only

one or two indexable fields, another tenant's data may include fewer, longer records having numerous indexable fields.

[0006] In addition to these structural (schema) differences, the distribution of data among different tenants may be quite different, even when their schemas are similar. Modern relational databases rely on statistics-based query optimizers that make decisions about the best manner to answer a query given accurate table-level and column-level statistics that are gathered periodically. Importantly, however, because existing relational databases are not multi-tenant aware, these statistics cut across all tenants in the database. That is, the statistics that are gathered are not specific to any one tenant, but are in fact an aggregate or average of all tenants. This approach can lead to incorrect assumptions and query plans about any one tenant.

[0007] As a specific example, Oracle provides a query optimizer that can be used on an Oracle database. This query optimizer works generally as follows: for each table, column, or index, aggregate statistics are gathered (typically periodically or on demand by a database administrator ("DBA")). The gathered statistics typically include the total number of rows, average size of rows, total number of distinct values in a column or index (an index can span multiple columns), histograms of column values (which place a range of values into buckets), *etc.* The optimizer then uses these statistics to decide among a possible set of data access paths.

[0008] In general, one goal of a query optimizer is to minimize the amount of data that must be read from disk (*e.g.*, because disk access may be a slow operation). The optimizer therefore typically chooses tables or columns that are most "selective" – that is, will yield the fewest rows when the query condition is evaluated. For instance, if a single query filters on two columns of a single table, and both columns are indexed, then the optimizer will use the index that has the highest number of distinct values because statistically for any given filter value a smaller number of rows are expected to be returned. If the optimizer knows that a certain column has a very high cardinality (number of distinct values) then the optimizer will choose to use an index on that column versus a similar index on a lower cardinality column. The optimizer assumes relatively even distribution of data and therefore reaches the conclusion that the high-cardinality column is likely to yield a smaller number of satisfying-rows for a given equality filter.

[0009] Now consider in a multi-tenant system a physical column (shared by many tenants) that has a large number of distinct values for most tenants, but a small number of distinct values for a specific tenant. For this latter tenant the query optimizer will use this overall-high-cardinality column in error – because the optimizer is unaware that for this specific tenant the column is not selective.

[0010] In the case of table joins, the optimizer's decisions may be even more important – deciding which table to retrieve first can have a profound impact on overall query performance. Here again, by using system-wide aggregate statistics the optimizer might choose a query plan that is incorrect or inefficient for a single tenant that does not conform to the “normal” average of the entire database as determined from the gathered statistics.

[0011] Accordingly, it is desirable to provide systems and methods for optimizing database queries, and for dynamically tuning a query optimizer, in a multi-tenant database system which overcome the above and other problems.

BRIEF SUMMARY OF THE INVENTION

[0012] The present invention provides methods and systems for optimizing database queries in a multi-tenant database system. In certain aspects, for example, the present invention provides methods for dynamically tuning a query optimizer based on particular data characteristics of the tenant whose data is being searched and the particular query being executed.

[0013] The systems and methods of the present invention advantageously harness greater semantic knowledge about the use of data tables by the underlying relational database. By tracking tenant-level statistics on top of the typical system-gathered statistics (e.g., Oracle-gathered statistics), the present invention is advantageously able to optimize queries and/or make recommendations to the underlying query optimizer to improve its knowledge of the data, and therefore increase system performance, particularly for individual tenants.

[0014] According to the present invention, a multi-tenant database stores data from multiple tenants. While the overall database structure or schema is fixed, each tenant may have a tenant-specific virtual schema that describes the logical structure of that tenant's data. In certain aspects, each tenant's virtual schema includes a variety of customizable fields, some or all of which may be designated as indexable.

[0015] According to an aspect of the present invention, [Redacted]

[0016] According to another aspect of the present invention, [Redacted]

5 [0017] Reference to the remaining portions of the specification, including the drawings and claims, will realize other features and advantages of the present invention. Further features and advantages of the present invention, as well as the structure and operation of various embodiments of the present invention, are described in detail below with respect to the accompanying drawings. In the drawings, like reference numbers indicate identical or
10 functionally similar elements.

BRIEF DESCRIPTION OF THE DRAWINGS

[0018] FIG. 1 illustrates an environment wherein a multi-tenant database system (MTS) might be used according to one embodiment.

15 [0019] FIG. 2 illustrates elements of an MTS and interconnections therein in more detail according to one embodiment.

[0020] FIG. 3 illustrates an example of a data model for sharing.

DETAILED DESCRIPTION OF THE INVENTION

20 [0021] FIG.1 illustrates an environment wherein a multi-tenant database system might be used. As illustrated in FIG. 1 (and in more detail in FIG. 2) any user systems 12 might interact via a network 14 with a multi-tenant database system (MTS) 16. The users of those user systems 12 might be users in differing capacities and the capacity of a particular user system 12 might be entirely determined by the current user. For example, where a
25 salesperson is using a particular user system 12 to interact with MTS 16, that user system has the capacities allotted to that salesperson. However, while an administrator is using that user system to interact with MTS 16, it has the capacities allotted to that administrator.

[0022] Network 14 can be a LAN (local area network), WAN (wide area network), wireless network, point-to-point network, star network, token ring network, hub network, or other
30 configuration. As the most common type of network in current use is a TCP/IP (Transfer Control Protocol and Internet Protocol) network such as the global internetwork of networks

often referred to as the “Internet” with a capital “I,” that will be used in many of the examples herein, but it should be understood that the networks that the present invention might use are not so limited, although TCP/IP is the currently preferred protocol.

[0023] User systems 12 might communicate with MTS 16 using TCP/IP and, at a higher network level, use other common Internet protocols to communicate, such as HTTP, FTP, AFS, WAP, *etc.* As an example, where HTTP is used, user system 12 might include an HTTP client commonly referred to as a “browser” for sending and receiving HTTP messages from an HTTP server at MTS 16. Such HTTP server might be implemented as the sole network interface between MTS 16 and network 14, but other techniques might be used as well or instead. In some implementations, the interface between MTS 16 and network 14 includes load sharing functionality, such as round-robin HTTP request distributors to balance loads and distribute incoming HTTP requests evenly over a plurality of servers. Preferably, each of the plurality of servers has access to the MTS’s data, at least as for the users that are accessing a server.

[0024] In preferred aspects, the system shown in FIG. 1 implements a web-based customer relationship management (CRM) system and/or a sales force automation (SFA) system. For example, in one aspect, MTS 16 can include application servers configured to implement and execute CRM and SFA software applications as well as provide related data, code, forms, web pages and other information to and from user systems 12 and to store to, and retrieve from, a database system related data, objects and web page content. With a multi-tenant system, tenant data is preferably arranged so that confidential information of one tenant is kept separate from another tenant’s data so that that other tenant does not have access to that confidential data.

[0025] One arrangement for elements of MTS 16 is shown in FIG. 1, including a network interface 20, storage 22 for tenant data, storage 24 for system data accessible to MTS 16 and possibly multiple tenants, program code 26 for implementing various functions of MTS 16, and a process space 28 for executing MTS system processes and tenant specific processes, such as running applications as part of an application service.

[0026] Several elements in the system shown in FIG. 1 include conventional, well-known elements that need not be explained in detail here. For example, each user system 12 could include a desktop personal computer, workstation, laptop, PDA, cell phone, or any WAP-enabled device or any other computing device capable of interfacing directly or indirectly to

the Internet or other network connection. User system 12 typically runs an HTTP client, *e.g.*, a browsing program, such as Microsoft's Internet Explorer™ browser, Netscape's Navigator™ browser, Opera's browser, or a WAP-enabled browser in the case of a cell phone, PDA or other wireless device, or the like, allowing a user (*e.g.*, subscriber of a CRM or SFA system) of user system 12 to access, process and view information and pages available to it from MTS 16 over network 14. Each user system 12 also typically includes one or more user interface devices, such as a keyboard, a mouse, touch screen, pen or the like, for interacting with a graphical user interface (GUI) provided by the browser on a display (*e.g.*, monitor screen, LCD display, *etc.*) in conjunction with pages, forms and other information provided by MTS 16 or other systems or servers. As discussed above, the present invention is suitable for use with the Internet, which refers to a specific global internetwork of networks. However, it should be understood that other networks can be used instead of the Internet, such as an intranet, an extranet, a virtual private network (VPN), a non-TCP/IP based network, any LAN or WAN or the like.

[0027] According to one embodiment, each user system 12 and all of its components are operator configurable using applications, such as a browser, including computer code run using a central processing unit such as an Intel Pentium processor or the like. Similarly, MTS 16 (and additional instances of MTS's, where more than one is present) and all of their components might be operator configurable using application(s) including computer code run using a central processing unit such as an Intel Pentium processor or the like, or multiple processor units. Computer code for operating and configuring MTS 16 to intercommunicate and to process web pages and other data and media content as described herein is preferably downloaded and stored on a hard disk, but the entire program code, or portions thereof, may also be stored in any other volatile or non-volatile memory medium or device as is well known, such as a ROM or RAM, or provided on any media capable of storing program code, such as a compact disk (CD) medium, digital versatile disk (DVD) medium, a floppy disk, and the like. Additionally, the entire program code, or portions thereof, may be transmitted and downloaded from a software source, *e.g.*, over the Internet, or from another server, as is well known, or transmitted over any other conventional network connection as is well known (*e.g.*, extranet, VPN, LAN, *etc.*) using any communication medium and protocols (*e.g.*, TCP/IP, HTTP, HTTPS, Ethernet, *etc.*) as are well known. It will also be appreciated that computer code for implementing aspects of the present invention can be implemented in any programming language that can be executed on a server or server system such as, for

example, in C, C+, HTML, Java, JavaScript, or any other scripting language, such as VBScript.

[0028] According to one embodiment, each MTS 16 is configured to provide web pages, forms, data and media content to user systems 12 to support the access by user systems 12 as tenants of MTS 16. As such, MTS 16 provides security mechanisms to keep each tenant's data separate unless the data is shared. If more than one MTS is used, they may be located in close proximity to one another (*e.g.*, in a server farm located in a single building or campus), or they may be distributed at locations remote from one another (*e.g.*, one or more servers located in city A and one or more servers located in city B). As used herein, MTS's could include one or more logically and/or physically connected servers distributed locally or across one or more geographic locations. Additionally, the term "server" is meant to include a computer system, including processing hardware and process space(s), and an associated storage system and database application as is well known in the art. It should also be understood that "server system" and "server" are often used interchangeably herein. Similarly, the databases described herein can be implemented as single databases, a distributed database, a collection of distributed databases, a database with redundant online or offline backups or other redundancies, *etc.*, and might include a distributed database or storage network and associated processing intelligence.

[0029] FIG. 2 illustrates elements of MTS 16 and various interconnections in more detail. In this example, the network interface is implemented as one or more HTTP application servers 100. Also shown is system process space 102 including individual tenant process spaces 104, a system database 106, tenant database(s) 108 and a tenant management process space 110. Tenant database 108 might be divided into individual tenant storage areas 112, which can be either a physical arrangement or a logical arrangement. Within each tenant storage area 112, user storage 114 might be allocated for each user.

[0030] It should also be understood that each application server 100 may be communicably coupled to database systems, *e.g.*, system database 106 and tenant database(s) 108, via a different network connection. For example, one server 100₁ might be coupled via the Internet 14, another server 100_{N-1} might be coupled via a direct network link, and another server 100_N might be coupled by yet a different network connection. Transfer Control Protocol and Internet Protocol (TCP/IP) are preferred protocols for communicating between servers 100 and the database system, however, it will be apparent to one skilled in the art that other

transport protocols may be used to optimize the system depending on the network interconnect used.

[0031] In preferred aspects, each application server 100 is configured to handle requests for any user/organization. Because it is desirable to be able to add and remove application servers from the server pool at any time for any reason, there is preferably no server affinity for a user and/or organization to a specific application server 100. In one embodiment, therefore, an interface system (not shown) implementing a load balancing function (*e.g.*, an F5 Big-IP load balancer) is communicably coupled between the servers 100 and the user systems 12 to distribute requests to the servers 100. In one aspect, the load balancer uses a least connections algorithm to route user requests to the servers 100. Other examples of load balancing algorithms, such as are round robin and observed response time, also can be used. For example, in certain aspects, three consecutive requests from the same user could hit three different servers, and three requests from different users could hit the same server. In this manner, MTS 16 is multi-tenant, wherein the MTS 16 handles storage of different objects and data across disparate users and organizations.

[0032] As an example of storage, one tenant might be a company that employs a sales force where each salesperson uses MTS 16 to manage their sales process. Thus, a user might maintain contact data, leads data, customer follow-up data, performance data, goals and progress data, all applicable to that user's personal sales process (*e.g.*, in tenant database 108). In the preferred MTS arrangement, since all of this data and the applications to access, view, modify, report, transmit, calculate, *etc.*, can be maintained and accessed by a user system having nothing more than network access, the user can manage his or her sales efforts and cycles from any of many different user systems. For example, if a salesperson is paying a visit to a customer and the customer has Internet access in their lobby, the salesperson can obtain critical updates as to that customer while waiting for the customer to arrive in the lobby.

[0033] While each user's sales data might be separate from other users' sales data regardless of the employers of each user, some data might be organization-wide data shared or accessible by a plurality or all of the sales force for a given organization that is a tenant. Thus, there might be some data structures managed by MTS 16 that are allocated at the tenant level while other data structures are managed at the user level. Because an MTS might support multiple tenants including possible competitors, the MTS should have security

protocols that keep data, applications and application use separate. Also, because many tenants will opt for access to an MTS rather than maintain their own system, redundancy, up-time and backup are more critical functions and need to be implemented in the MTS.

[0034] In addition to user-specific data and tenant-specific data, MTS 16 might also maintain system level data usable by multiple tenants. Such system level data might include industry reports, news, postings, and the like that are sharable among tenants.

[0035] In certain aspects, client systems 12 communicate with application servers 100 to request and update system-level and tenant-level data from MTS 16 that may require one or more queries to database system 106 and/or database system 108. MTS 16 generates automatically one or more SQL statements (the SQL query) designed to access the desired information.

[0036] Each database can generally be viewed as a set of logical tables containing data fitted into predefined categories. Each table generally contains one or more data categories logically arranged in physical columns. Each row of a table contains an instance of data for each category defined by the columns. For example, a CRM database may include a table that describes a customer with columns for basic contact information such as name, address, phone number, fax number, *etc.* Another table might describe a purchase order, including columns for information such as customer, product, sale price, date, *etc.*

[0037] Now, consider in a multi-tenant system a physical column (shared by many tenants) that has a large number of distinct values for most tenants, but a small number of distinct values for a specific tenant. For this latter tenant, a typical database optimizer will choose to use this overall-high-cardinality column in error because the optimizer is unaware that for this specific tenant the column is not selective.

[0038] In the case of table joins the optimizer's decisions may be even more important – deciding which table to retrieve first can have a profound impact on overall query performance. Here again, by using system-wide aggregate statistics the optimizer might choose a query plan that is incorrect or inefficient for a single tenant that does not conform to the “normal” average of the entire database.

[0039] As a specific example of the importance of table joins, consider the sharing feature in the Salesforce.com service. The sharing feature allows a specific list of users to have access to privileged data, such as specific accounts or opportunities. In one aspect, a Many-

to-Many (MTM) physical table serves as the storage for this sharing information. The MTM table specifies that a user has access to a particular entity (*e.g.*, account or opportunity) row. When displaying a list of all rows that the current user can see (possibly with a filter on the entity rows, such as the name of the account or the dollar amount of the opportunity) the query optimizer must choose between accessing this MTM table from the user or the entity side of the relationship. If the entity filter is highly selective (for instance, a particular account name such as “XYZ Corp”) it will generally make more sense to begin the query access path from this side. If, however, the entity is not filtered selectively, but the current user has access to a small amount of data, then the query optimizer should access rows in the MTM table through the user side of this relationship.

[0040] However, in the above example in a multi-tenant database system the optimizer’s native statistics may be insufficient to make this determination because the native statistics will aggregate across too many tenants and will not have context into the current tenant’s data.

[0041] Note, because of the wide range of business types, industries, and sizes potentially served by multi-tenant database systems such as the Salesforce.com service, the likelihood of data “skew” is greatly increased. That is, the statistical profile of the largest most complex tenants is likely to look very different from that of small or medium sized customers.

[0042] In Oracle database systems, override mechanisms are provided to affect the Oracle automatic query optimizer. The use of query “Hints” allows the SQL author the ability to choose explicitly a query plan. For instance, a human-authored SQL statement might mention the explicit order of table joins, or explicit index names to use (rather than letting the optimizer choose automatically). Another mechanism for controlling the query plan explicitly is to re-write the query using different SQL syntax. For instance, a single flat SQL statement can be re-written using a nested SELECT in the FROM clause of the outer query. Joins and semi-joins are sometimes inter-changeable. Anti-joins can be written using the MINUS operator, *etc.* All of these are examples of ways in which a human-author, or a programmatic SQL generator, can alter the behavior of the underlying query optimizer by using higher-level knowledge to change the plan.

[0043] In certain aspects, the present invention configures or tunes a query optimizer, such as the Oracle query optimizer, by supplying appropriate “hints.” For example, when SQL is generated programmatically by the MTS, the tenant-level statistics are consulted and a

dynamic decision is made as to the syntax of the query. The tenant level statistics preferably mirror the underlying relational database statistics in many ways (for example, in one aspect they track the total number of distinct values for indexed columns) but the statistics are kept on a per-tenant basis (*e.g.*, in tables in tenant database storage areas 112). Similarly for
5 important application functionality, such as the sharing feature, the MTS tracks the approximate number of rows to which each user has access and stores such statistics (*e.g.*, tables stored in user storage areas 114 of database 108). Then, when a filtered sharing query arrives, the dynamically generated SQL includes the appropriate hints and structure to force a query plan that is optimal.

10 [0044] Optimizer

[0045] In one aspect, metadata information about users and organizations and the data contained in entity rows for that organization are tracked [Redacted]

in

15 order to make choices about query access paths, particularly for list-style queries such as reports. The areas targeted include:

1. The evaluation of a sharing model, which controls which users can see which records. The optimizer preferably distinguishes between users that can see many rows in an organization (*e.g.*, bosses) versus users who can see very few rows (*e.g.*,
20 lower level employees).
2. The choice of which filters are the most selective for fields that contain enumerated lists of values (*e.g.*, list of status values for an account, list of industries, list of states, *etc.*).

[0046] Sharing model

25 [0047] For each user in the system an approximate count of the number of rows (for each entity type that has a sharing model) that the user can see is tracked. This number (as a percentage of the total number of entity rows for that organization) is used as a decision point in choosing between two different query paths. It has been determined empirically that users who can see most of the entity rows (*e.g.*, bosses) benefit from a certain query structure,
30 whereas users who can see a small percentage of the entity rows (*e.g.*, lower level employees) benefit from a different query structure. Current systems are not able to choose between these paths without having an entirely different SQL provided via a programmatic decision.

[0048] In order to keep the statistics up to date it is important to track the percentage of rows that each and every user can see. In one aspect, there are three ways in which a user might gain access to data in a private security model:

- 5 (1) Rows owned by the user or users below him in the role hierarchy;
- (2) Rows that are shared via sharing rules to a group to which this user belongs; and
- (3) Rows that are shared via manual/team sharing to this user (possibly via a group).

[0049] In a preferred aspect, statistics and metadata are tracked for user and organization
10 quotas. In some aspects, such information is tracked periodically (*e.g.*, on a scheduled basis – during off-peak hours, amortizing the work over multiple days), wherein the number of visible rows for each user is calculated exactly or approximately, or before every Nth query (*e.g.*, every 10th query) by a user, that user's visibility is calculated explicitly and then that
15 statistic is used until it is again calculated (here it is assumed that users do not change very often from one strategy to another). In yet a further aspect, whenever an unconstrained query is run, the number of visible rows is remembered and that number is used until the user runs the next unconstrained query.

[0050] In one aspect, the total number of rows for each entity type for each organization is tracked (this will be useful for any of the strategies above). Also the total number of rows
20 owned by each user in a metadata table is tracked.

[0051] If it is assumed that (1) and (2) are the most important reasons for why a user has access to entity records (this might be known empirically from how organizations use the system) then the information needed to calculate the number of rows a user can see, approximately, is known. Namely, the role hierarchy metadata tables can be used in
25 conjunction with the metadata table to determine the number of records owned by the user or his subordinates. The sharing rule metadata can also be used along with the group definition metadata to calculate the total number of rows visible via sharing rules. While these two sets may overlap, for the purpose of the heuristic decision between “boss” and “lower level employee,” the sum of these two values is sufficiently close to the true value.

30 [0052] In one aspect, the use of metadata tables only (which are generally much smaller than the actual entity tables which might have millions of rows) ensures that the calculation of visible rows will itself not require much time. In one aspect, this decision is cached in a user-information data structure kept by the running application servers 100, *e.g.*, with a

timeout value. In this manner, even though the calculation itself may be relatively lightweight, it is only performed periodically while a user is logged in.

[0053] To focus on how the “boss” vs “lower level employee” decision should drive an appropriate query plan, consider a query of the form: “Show me all accounts that I can see” in a private account sharing model. An example of a data model for sharing appears in FIG. 3 (middle table is sharing table, final table is the user/group “blowout” table which describes which users are contained in a group, or above a user in the role hierarchy (UG = User or Group)). According to one aspect, for a “lower level employee” user it is typically most advantageous to join these tables starting from the right, filtering on users Id to form a temporary result of the rows that can be seen. Because the user can not see many rows, this will yield a relatively selective path. An example query follows:

```
select a.name "ACCOUNT.NAME",
from sales.account a,
(select distinct s.account_id
from core.ug_blowout b, sales.acc_share s
    where s.organization_id = ?
    and b.organization_id = ?
    and b.users_id = ?
    and s.ug_id = b.ug_id
    and s.acc_access_level > 0) t,
core.users u
where (t.account_id = a.account_id)
and (u.users_id = a.owner)
and (a.deleted = '0')
and (a.organization_id = ?)
and (u.organization_id = ?))
```

[0054] Conversely for a “boss” user who can see most of the entity records in the organization, it is typically most advantageous to begin the query from the left and use a nested loop query plan onto the sharing table (acc_share), an example of which follows:

```
select
    a.name "ACCOUNT.NAME",
from
    sales.account a, core.users u
where (u.users_id = a.owner)
and (a.deleted = '0')
and (a.organization_id = ?)
and (exists (select 1
    from core.ug_blowout b,
    sales.acc_share s
    where s.organization_id = ?
```

```

        and b.organization_id = ?
        and b.users_id = ?
        and s.ug_id = b.ug_id
        and s.acc_access_level > 0
5      and s.account_id = a.account_id))
    and (u.organization_id = ?)

```

[0055] Note that this query in general runs in relatively constant (reasonable) time for all users in an organization. It may not be particularly fast since it must look at all top-level entity records, but it is suitable for a boss who can in fact see most records. The first “lower level employee” query runs much faster for users who in fact can not see many records, but it may run much slower for bosses who can see all records. This, again, is why it is desirable to have an accurate decision between the two paths.

[0056] *Filter Choice*

[0057] A typical end user report execution includes a set of displayed columns from multiple tables along with a set of filter conditions. A typical report might join between 3 and 7 (or more) main tables with filtering possibly occurring on one or more of these tables. In addition, certain filters, such as the sharing filter discussed above (which can take the form of an additional join or a nested sub-query), should be applied to assure that the end user only sees data to which he has been given access.

[0058] Information about enumerated “picklist” fields (those fields that are known to contain a small list of possible values) are tracked in one aspect. Examples of these fields include the list of priority values for a task and the list of industries for an account. These fields are often used as filters for executive reporting and data rollup reports. In addition to the values themselves, the approximate number of times each value appears in the actual entity table for that organization (tenant) is tracked in the metadata. When a user provides a filter value such that the value appears infrequently for that organization, the overall query is preferably driven from that table and possibly from an index on that column, if such as index exists.

[0059] In one aspect, when a user runs a report with N filters, each filter is evaluated for expected selectiveness. If, for example, the user filters on “California” and “Florida” from a list of states and it is known that these values represent, respectively, 5 and 2 percent of the overall rows, then it is assumed that the filter has a 7% selectivity. Similarly if a boolean field has 95% true values, then filtering on false appears attractive as a filter, whereas

filtering on Male from a random list of people would not be very selective, since 50% reduction would not make a good filter condition.

[0060] The selectivity of the sharing condition is also considered in one aspect. For a user with very low (perhaps 2%) visibility, the sharing filter might prove to be the best starting point and therefore the optimizer is instructed to begin with the filter, rather than one of the main entity tables such as, *e.g.*, Account or Contact tables.

[0061] The cost-based optimizer, in one aspect, incorporates other filter types, in addition to semantics knowledge about the application. For example, if an organization has imported all opportunities for the last 3 years, and a user report filters on “all opportunities that closed in the last week” then this is likely to represent a selective filter. The presence of custom field indexes (*e.g.*, a certain set of columns that administrators can choose to place into a B-tree indexed custom field columns into these heuristic decisions) are also factored in one aspect.

[0062] In one aspect, a query is only hinted if it is assumed that a particular starting table will yield a selective path. All other tables would then be joined via nested loops. Note, these types of cost-based decisions are similar to the decisions that the optimizer (*e.g.*, Oracle optimizer) itself makes when deciding how to join tables. Importantly, the system of the present invention makes tenant-level data decisions based on tenant-level statistics and user-level data decisions based on user-level statistics. The system of the present invention also takes into account application-level concepts such as sharing that are beyond the generic nature of the underlying RBDMS.

[0063] For picklist fields, the statistics tracked and stored do not need to reflect the exact number of occurrences for each value, a reasonable estimate is sufficient in one aspect. Values missing from the statistics either do not occur at all, or occur infrequently – it is assumed that they make good filters.

[0064] In one embodiment, each update or insert into an entity table passes through an application server 100. Therefore as the data is being processed in Java counters are incremented and decremented for individual picklist values. Inserts increment a counter, while updates to a different value decrement a counter for the old value and increment a counter for the new value. Since these statistics do not need to be exact, the statistics metadata is preferably not updated with each and every database insert or update (which might affect performance). Rather, in one aspect, an in-memory cache server (which already

contains the metadata for valid picklist values) is augmented with the counters for these values, with the ability to update the database values periodically to persist the changes. An example of such a cache server can be found in U.S. Patent Application Serial No.

10/418,961, filed April 17, 2003, the contents of which are hereby incorporated by reference

5 in its entirety.

[0065] For row deletion, the data preferably does not pass through the application server

100. However, all main entities are preferably soft-deleted (with a modification stamp),

meaning the rows are left in the database for possible un-deletion. Therefore, an

asynchronous process is preferably used to update the statistics when rows are deleted and

10 un-deleted since it is known which rows have been touched since the last running of that

process.

[0066] While the invention has been described by way of example and in terms of the

specific embodiments, it is to be understood that the invention is not limited to the disclosed

embodiments. To the contrary, it is intended to cover various modifications and similar

15 arrangements as would be apparent to those skilled in the art. Therefore, the scope of the

appended claims should be accorded the broadest interpretation so as to encompass all such

modifications and similar arrangements.

WHAT IS CLAIMED IS:

1 1. A method of optimizing a query in a multi-tenant database, said
2 database having one or more data tables, each table having one or more logical columns
3 defining data categories and one or more logical rows associated with one or more tenants,
4 wherein a plurality of tenants have data stored in the data tables, the method comprising:
5 generating tenant-level statistics for each of said plurality of tenants for each
6 of the data tables;
7 receiving a SQL query; and
8 optimizing the SQL query based on the tenant-level statistics.

1 2. The method of claim 1, wherein each tenant includes one or more
2 associated users, the method further including:
3 generating user-level statistics for each user of each tenant for each of the data
4 tables; and
5 optimizing the SQL query based on the user-level statistics.

1 3. The method of claim 1, wherein at least one column of one table
2 includes data associated with two or more tenants.

1 4. The method of claim 1, wherein generating includes **[Redacted]**
2

1 5. A multi-tenant database system, comprising:
2 a database having one or more data tables, each table having one or more
3 columns defining data categories and one or more rows associated with one or more tenants,
4 wherein a plurality of tenants have data stored in the data tables;
5 a statistics generating module configured to generate tenant-level statistics for
6 each tenant for each of the data tables; and
7 a query optimization module, configured to optimize a database query based
8 on the tenant-level statistics.

1 6. The multi-tenant database system of claim 5, wherein each tenant
2 includes one or more associated users, wherein the statistics generating module is further
3 configured to generate user-level statistics for each user, and wherein the query optimization
4 module is further configured to optimize the database query based on the user-level statistics.

- 1 7. The system of claim 1, wherein the statistic generation module **[Redacted]**
2
- 1 8 **[more dependent claims to follow...].**

QUERY OPTIMIZATION IN A MULTI-TENANT DATABASE SYSTEM

ABSTRACT OF THE DISCLOSURE

More efficient querying of a multi-tenant database using dynamic tuning of database indices. A layer of meta-data associates data items with tenants, *e.g.*, via tags, and the meta-data is used to optimize searches by channeling processing resources during a query to only those pieces of data bearing the relevant tenant's unique tag.

59055939 v1